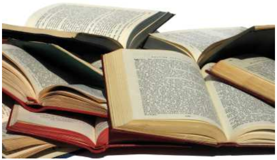
Provable Learning of Feature Representations

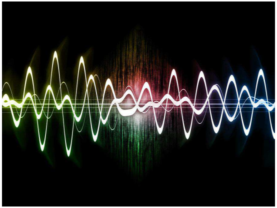
Anima Anandkumar

U.C. Irvine

Feature learning as cornerstone of ML

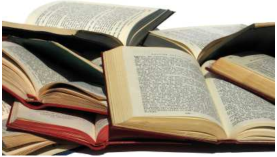
ML Practice

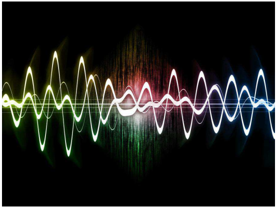




Feature learning as cornerstone of ML

ML Practice





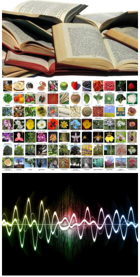
ML Papers

Label	Features			
0	2.1	5.2	0	0
1	0	0	2	1
1	1.1	0	0	0
0	0	0	7	0

Feature learning as cornerstone of ML

- Find efficient representation of data, e.g. based on sparsity, low dimensional structures etc.

ML Practice



ML Papers

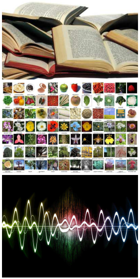
Label	Features				
0	2.1	5.2	0	0	...
1	0	0	2	1	...
1	1.1	0	0	0	...
0	0	0	7	0	...
⋮	⋮	⋮	⋮	⋮	

- Feature engineering typically critical for good performance
- Deep learning has shown considerable promise for feature learning

Feature learning as cornerstone of ML

- Find efficient representation of data, e.g. based on sparsity, low dimensional structures etc.

ML Practice



ML Papers

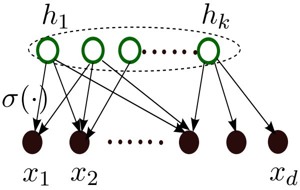
Label	Features				
0	2.1	5.2	0	0	...
1	0	0	2	1	...
1	1.1	0	0	0	...
0	0	0	7	0	...
⋮	⋮	⋮	⋮	⋮	

- Feature engineering typically critical for good performance
- Deep learning has shown considerable promise for feature learning
- **Can we provide principled approaches which are guaranteed to learn good features?**

Neural Networks and Unsupervised Learning

Belief Networks/Boltzmann Machines

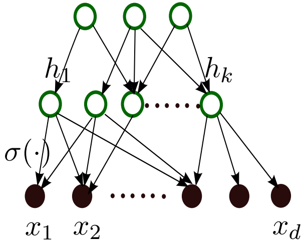
- Observation: $x = \sigma(Ah + b)$, where $\sigma(\cdot)$ is any (non-linear) function.
- $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^k$.
- Unsupervised setting: h is unobserved.
- Deep networks: $\sigma(\cdot)$ applied recursively.
- Probabilistic model: $\mathbb{E}[x|h] = \sigma(Ah + b)$.



Neural Networks and Unsupervised Learning

Belief Networks/Boltzmann Machines

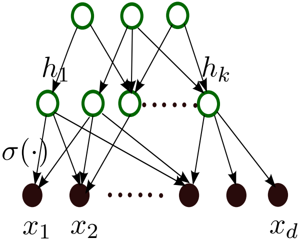
- Observation: $x = \sigma(Ah + b)$, where $\sigma(\cdot)$ is any (non-linear) function.
- $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^k$.
- Unsupervised setting: h is unobserved.
- Deep networks: $\sigma(\cdot)$ applied recursively.
- Probabilistic model: $\mathbb{E}[x|h] = \sigma(Ah + b)$.



Neural Networks and Unsupervised Learning

Belief Networks/Boltzmann Machines

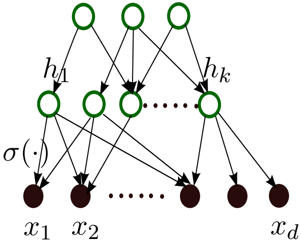
- Observation: $x = \sigma(Ah + b)$, where $\sigma(\cdot)$ is any (non-linear) function.
- $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^k$.
- Unsupervised setting: h is unobserved.
- Deep networks: $\sigma(\cdot)$ applied recursively.
- Probabilistic model: $\mathbb{E}[x|h] = \sigma(Ah + b)$.



In this talk: for simplicity, noiseless case. Most analysis carries over.

Neural Networks and Unsupervised Learning

Belief Networks/Boltzmann Machines



- Observation: $x = \sigma(Ah + b)$, where $\sigma(\cdot)$ is any (non-linear) function.
- $x \in \mathbb{R}^d$ and $h \in \mathbb{R}^k$.
- Unsupervised setting: h is unobserved.
- Deep networks: $\sigma(\cdot)$ applied recursively.
- Probabilistic model: $\mathbb{E}[x|h] = \sigma(Ah + b)$.

In this talk: for simplicity, noiseless case. Most analysis carries over.

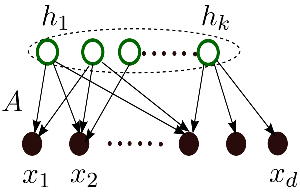
Learning

- Through gradient descent.
- Non-convex: no guarantees in general.

In this talk: methods and guarantees for learning neural networks

Linear Neural Networks

- Observed sample $x = Ah$.
- h is hidden variable and A is dictionary.
- $x \in \mathbb{R}^d$, $h \in \mathbb{R}^k$ and $A \in \mathbb{R}^{d \times k}$.

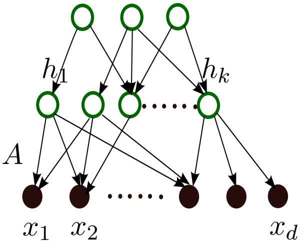


Observations

- Deep networks can be collapsed: 1-layer networks suffice.
- Poor performance in practice.
- Natural to first analyze linear models.

Linear Neural Networks

- Observed sample $x = Ah$.
- h is hidden variable and A is dictionary.
- $x \in \mathbb{R}^d$, $h \in \mathbb{R}^k$ and $A \in \mathbb{R}^{d \times k}$.

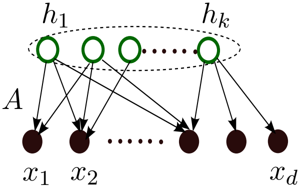


Observations

- Deep networks can be collapsed: 1-layer networks suffice.
- Poor performance in practice.
- Natural to first analyze linear models.

Linear Neural Networks

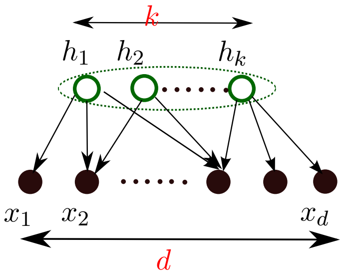
- Observed sample $x = Ah$.
- h is hidden variable and A is dictionary.
- $x \in \mathbb{R}^d$, $h \in \mathbb{R}^k$ and $A \in \mathbb{R}^{d \times k}$.



Observations

- Deep networks can be collapsed: 1-layer networks suffice.
- Poor performance in practice.
- Natural to first analyze linear models.

Learning Linear Networks through SVD



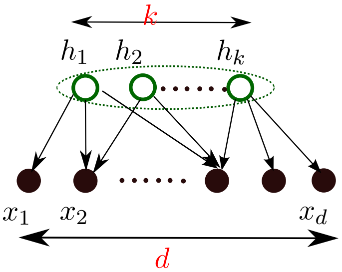
- Linear model: $x = Ah$.
- Pairwise moments: $M_2 = \mathbb{E}[xx^\top] = A\mathbb{E}[hh^\top]A^\top$.
- SVD: $M_2 = U\Lambda U^\top$: a valid linear representation.

- Learning through SVD: cannot learn overcomplete representations. $(k > d)$ learnable?
- SVD cannot enforce sparsity, non-negativity etc.

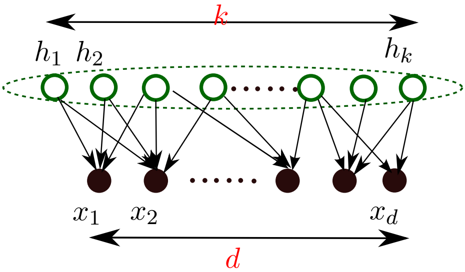
Learning Overcomplete Representations

- Latent dimensionality k and observed dimensionality d.

Undercomplete Representation



Overcomplete Representation



Works Analyzing Learning Linear Networks

In deep learning community

- No local optima for SVD: Baldi and Hornik '89.
- Dynamics of learning linear networks: Saxe et al '13.
- Undercomplete case and learning SVD representations.

In learning theory community (undercomplete models)

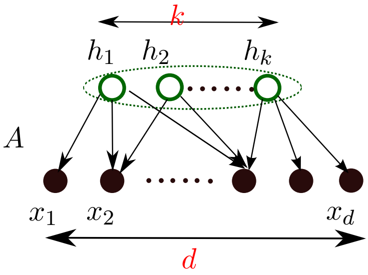
- Sparse representations: Spielman et. al'12, Anandkumar et. al'13.
- Non-negativity (topic modeling): Arora et. al. '12.
- Dirichlet models (LDA topic models): Anandkumar et. al. '12.

In learning theory community (overcomplete models)

- Concurrent works of Arora et. al. and Anandkumar et. al. for sparse coding
- Non-linear sparse representations: Arora et. al. '14.
- Overcomplete latent variable models: Anandkumar et. al. '14.

Outline

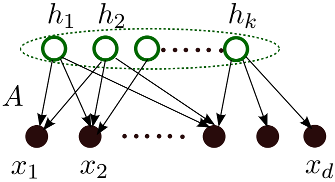
Learning Linear Sparse Representations



- Linear Model: $x = Ah$.
- Sparse representation: A is sparse.
- SVD need not give rise to sparse representations.

Guaranteed methods for learning sparse representations

Intuitions..

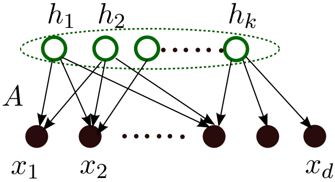


Learning using second-order moments

- Linear model: $\boxed{x = A\,h.}$ and $\boxed{\mathbb{E}[xx^\top] = A\mathbb{E}[hh^\top]A^\top}$
- Learning: recover A from $A\mathbb{E}[hh^\top]A^\top$.

Ill-posed without further restrictions

Intuitions..



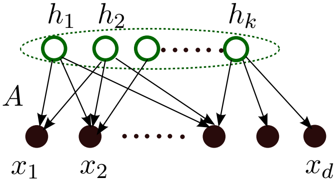
Learning using second-order moments

- Linear model: $x = A\,h.$ and $\mathbb{E}[xx^\top] = A\mathbb{E}[hh^\top]A^\top$
- Learning: recover A from $A\mathbb{E}[hh^\top]A^\top$.

Ill-posed without further restrictions

- When h is not degenerate: recover A from Col(A)
- Can we recover a sparse A?

Intuitions..



Learning using second-order moments

- Linear model: $x = A\,h.$ and $\mathbb{E}[xx^\top] = A\mathbb{E}[hh^\top]A^\top$
- Learning: recover A from $A\mathbb{E}[hh^\top]A^\top$.

Ill-posed without further restrictions

- When h is not degenerate: recover A from Col(A)
- Can we recover a sparse A?

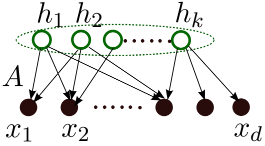
Sparsity constraints on topic-word matrix A

- Main constraint: columns of A are sparsest vectors in Col(A)

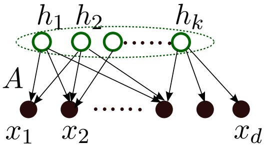
Sufficient Conditions for Identifiability

columns of A are sparsest vectors in Col(A)

- Sufficient conditions?



Sufficient Conditions for Identifiability



columns of A are sparsest vectors in Col(A)

- Sufficient conditions?

Structural Condition: (Additive) Graph Expansion

$|\mathcal{N}(S)| \geq |S| + d_{\max}$, for all $S \subset [k]$

Parametric Conditions: Generic Parameters

$$\|Av\|_0 > |\mathcal{N}_A(\text{supp}(v))| - |\text{supp}(v)|$$

Tractable Algorithm for Unmixing

Unmixing Task

Recover topic-word matrix A from $\boxed{A\mathbb{E}[hh^\top]A^\top}$.

Exhaustive search

$$\boxed{\min_{z \neq 0} \|Az\|_0}$$

Convex relaxation

$$\boxed{\min_z \|Az\|_1, \quad b^\top z = 1,}$$

where b is a row in A.

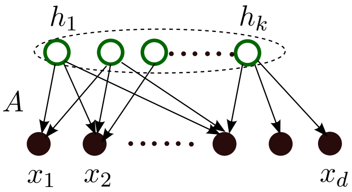
Change of Variables

$$\boxed{\min_w \|(A\mathbb{E}[hh^\top]A^\top)^{1/2} w\|_1, \quad e_i^\top (A\mathbb{E}[hh^\top]A^\top)^{1/2} w = 1.}$$

Under "reasonable" conditions, the above program exactly recovers A

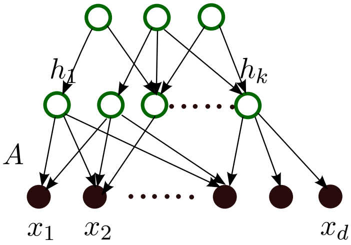
"Learning Latent Bayesian Networks and Topic Models Under Expansion Constraints" by A. Anandkumar, D. Hsu, A. Javanmard and S.M. Kakade. ICML, June 2013.

Learning Hierarchical Sparse Representations



- So far: recover topic-word matrix A from $\boxed{A\mathbb{E}[hh^\top]A^\top}$.
- Repeat the process to obtain hierarchical models.

Learning Hierarchical Sparse Representations

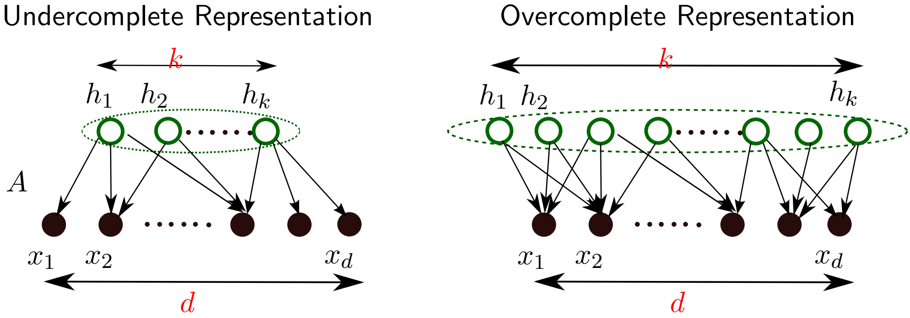


- So far: recover topic-word matrix A from $\boxed{A\mathbb{E}[hh^\top]A^\top}$.
- Repeat the process to obtain hierarchical models.

Outline

Learning Overcomplete Representations

- Latent dimensionality k and observed dimensionality d.



When are overcomplete models $(k > d)$ learnable?

Dictionary Learning or Sparse Coding

- Each sample is a sparse combination of dictionary atoms.

Dictionary Learning or Sparse Coding

- Each sample is a sparse combination of dictionary atoms.

Setup

- No. of dictionary elements $k >$ observed dimensionality d.
- Linear model: $X = AH$.
- $A = [a_1, \ldots, a_k]$: dictionary elements
- $x \in \mathbb{R}^d$: Observation. $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$: Observation matrix.

Main Assumptions

- H is sparse: each column is randomly s-sparse

  Each sample is a combination of s dictionary atoms.

Dictionary Learning or Sparse Coding

- Each sample is a sparse combination of dictionary atoms.

Setup

- No. of dictionary elements $k >$ observed dimensionality d.
- Linear model: $X = AH$.
- $A = [a_1, \ldots, a_k]$: dictionary elements
- $x \in \mathbb{R}^d$: Observation. $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$: Observation matrix.

Main Assumptions

- H is sparse: each column is randomly s-sparse

  Each sample is a combination of s dictionary atoms.
- A is incoherent: $\max_{i \neq j} |\langle a_i, a_j \rangle| \approx 0$.

Intuitions: how incoherence helps

- Each sample is a combination of dictionary atoms: $x_i = \sum_j h_{i,j} a_j$.
- Consider x_i and x_j s.t. they have no common dictionary atoms.
- What about $|\langle x_i, x_j \rangle|$?

Intuitions: how incoherence helps

- Each sample is a combination of dictionary atoms: $x_i = \sum_j h_{i,j} a_j$.
- Consider x_i and x_j s.t. they have no common dictionary atoms.
- What about $|\langle x_i, x_j \rangle|$?
- Under incoherence: $|\langle x_i, x_j \rangle| \approx 0$.

Intuitions: how incoherence helps

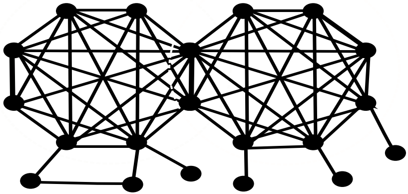
- Each sample is a combination of dictionary atoms: $x_i = \sum_j h_{i,j} a_j$.
- Consider x_i and x_j s.t. they have no common dictionary atoms.
- What about $|\langle x_i, x_j \rangle|$?
- Under incoherence: $|\langle x_i, x_j \rangle| \approx 0$.

Construction of Correlation Graph

- Nodes: Samples $x_1, \ldots, x_n$.
- Edges: $|\langle x_i, x_j \rangle| > \tau$ for some threshold τ.

How does the correlation graph help in dictionary learning?

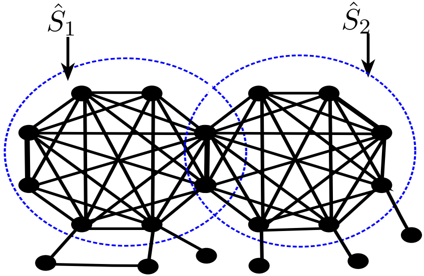
Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.

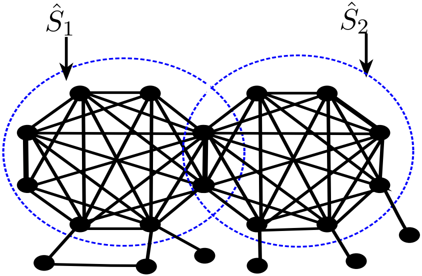
Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.

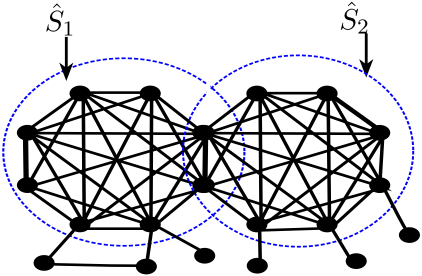
Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.
- Consider a large clique: a large fraction of pairs have exactly one element in common.

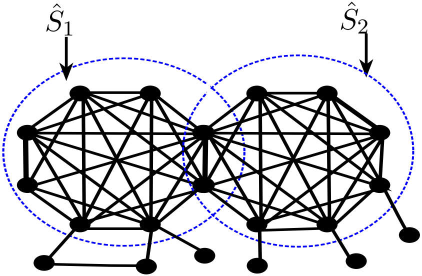
Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.
- Consider a large clique: a large fraction of pairs have exactly one element in common.
- How to find such a large clique efficiently?

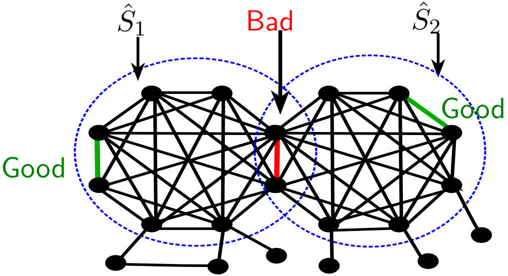
Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.
- Consider a large clique: a large fraction of pairs have exactly one element in common.
- How to find such a large clique efficiently? Start with a random edge.

Correlation Graph and Clique Finding



Main Insight

- (x_i, x_j): edge in correlation graph $\Rightarrow$ x_i and x_j have at least one dictionary element in common.
- Consider a large clique: a large fraction of pairs have exactly one element in common.
- How to find such a large clique efficiently? Start with a random edge.

Result on Approximate Dictionary Estimation

Procedure

- Start with a random edge (x_{i^*}, x_{j^*}).
- $\hat{S} =$ common nbd. of x_{i^*} and x_{j^*}. If $\hat{S}$ is close to a clique, accept.
- Estimate a dictionary element via top singular vector of $\sum_{i \in \hat{S}} x_i x_i^\top$.

Theorem

The dictionary A can be estimated with bounded error w.h.p. when $s = o(k^{1/3})$ and number of samples $n = \omega(k)$.

- Exact estimation when H is discrete, e.g. Bernoulli.

A. Agarwal, A., P. Netrapalli. "Exact Recovery of Sparsely Used Overcomplete Dictionaries," Preprint, Sept. 2013.

Exact Estimation via Alternating Minimization

- So far.. approximate dictionary estimation. What about exact estimation for arbitrary H?

Exact Estimation via Alternating Minimization

- So far.. approximate dictionary estimation. What about exact estimation for arbitrary H?

Alternating Minimization

- Given $X = AH$, initialize an estimate for A.
- Update H via ℓ_1 optimization.
- Re-estimate A via Least Squares.

Exact Estimation via Alternating Minimization

- So far.. approximate dictionary estimation. What about exact estimation for arbitrary H?

Alternating Minimization

- Given $X = AH$, initialize an estimate for A.
- Update H via ℓ_1 optimization.
- Re-estimate A via Least Squares.

- In general, alternating minimization converges to a local optimum.

Exact Estimation via Alternating Minimization

- So far.. approximate dictionary estimation. What about exact estimation for arbitrary H?

Alternating Minimization

- Given $X = AH$, initialize an estimate for A.
- Update H via ℓ_1 optimization.
- Re-estimate A via Least Squares.

- In general, alternating minimization converges to a local optimum.

Specific Initialization: Through our previous method.

Exact Estimation via Alternating Minimization

- So far.. approximate dictionary estimation. What about exact estimation for arbitrary H?

Alternating Minimization

- Given $X = AH$, initialize an estimate for A.
- Update H via ℓ_1 optimization.
- Re-estimate A via Least Squares.

- In general, alternating minimization converges to a local optimum.

Specific Initialization: Through our previous method.

Theorem

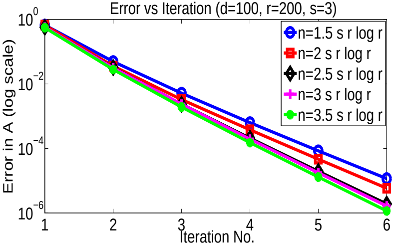
The above method converges to the true solution (A, H) at a linear rate w.h.p. when $s < \min(k^{1/8}, d^{1/9})$ and number of samples $n = \Omega(k^2)$.

A. Agarwal, A., P. Netrapalli, P. Jain, R. Tandon. "Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization," Preprint, Oct. 2013.

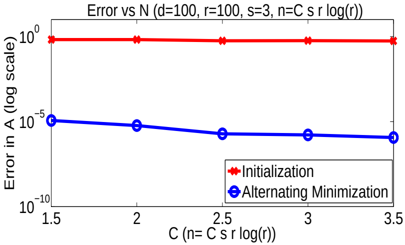
Outline

Simulations

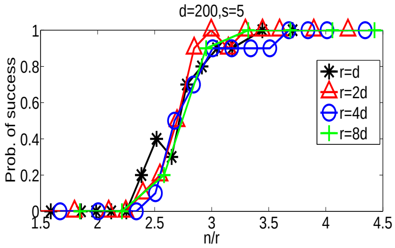
Local linear convergence



One-shot vs alternating

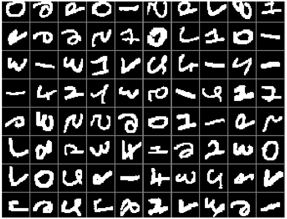


Sample complexity

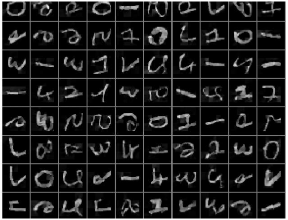


Experiments on MNIST

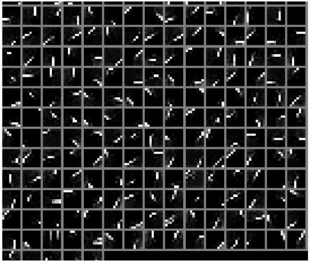
Original



Reconstruction



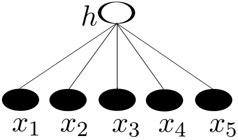
Learnt Representation



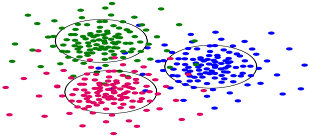
Outline

Tensor Methods for Unsupervised Learning

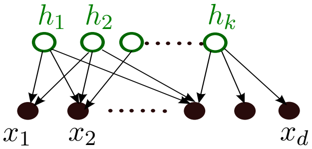
Multi-view mixtures



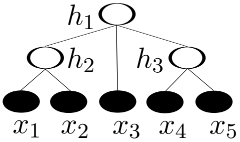
Spherical Gaussian mixtures



Indep. Component Analysis



HMM/Latent Trees



- Talk at spectral learning workshop at 15:40 today.

Conclusion

Learning Feature Representations

- Guaranteed unsupervised learning is possible in many cases
- Exploit availability of large number of unlabelled samples
- Overcomplete models provide flexibility in modeling, robust to noise

Learning Linear Networks (Undercomplete)

- Learning under expansion. Guaranteed learning through ℓ_1.

Learning Linear Networks (Overcomplete)

- Each sample is a sparse combination of dictionary atoms.
- Guarantees through clique finding and alternating minimization.

Outlook

- Extend guarantees to non-linear setting.
- Representational power of such networks.